OpenCL™ Device - Intel® Processor Graphics

Reference Manual

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804

Contents

Legal Information

By using this document, in addition to any agreements you have with Intel, you accept the terms set forth below.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

A "Mission Critical Application" is any application in which failure of the Intel Product could result, directly or indirectly, in personal injury or death. SHOULD YOU PURCHASE OR USE INTEL'S PRODUCTS FOR ANY SUCH MISSION CRITICAL APPLICATION, YOU SHALL INDEMNIFY AND HOLD INTEL AND ITS SUBSIDIARIES, SUBCONTRACTORS AND AFFILIATES, AND THE DIRECTORS, OFFICERS, AND EMPLOYEES OF EACH, HARMLESS AGAINST ALL CLAIMS COSTS, DAMAGES, AND EXPENSES AND REASONABLE ATTORNEYS' FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PRODUCT LIABILITY, PERSONAL INJURY, OR DEATH ARISING IN ANY WAY OUT OF SUCH MISSION CRITICAL APPLICATION, WHETHER OR NOT INTEL OR ITS SUBCONTRACTOR WAS NEGLIGENT IN THE DESIGN, MANUFACTURE, OR WARNING OF THE INTEL PRODUCT OR ANY OF ITS PARTS.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to:

http://www.intel.com/design/literature.htm.

Intel processor numbers are not a measure of performance. Processor numbers differentiate features within each processor family, not across different processor families. Go to: http://www.intel.com/products/processor_number/.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

Getting Help and Support

To get support, visit the product support forum at http://software.intel.com/en-us/forums/intel-opencl-sdk/.

For information on the OpenCL™ Runtime requirements, known issues and limitations, refer to the Release Notes.

What is New

This Reference Manual is the new document that dissociates from the *Intel® SDK for OpenCL™ Applications - User Manual* the information on the Intel OpenCL™ Runtime for Intel® Graphics. The document comprises the following:

- Information on the Intel OpenCL platforms and supported hardware
- Guidelines on using OpenCL Installable Client Driver
- Supported features and extensions on the Intel GPU and CPU OpenCL devices
- Notes on the implemented OpenCL optional features
- Image formats supported on Intel CPU and GPU OpenCL devices
- Build and linking options

See Also

Intel® SDK for OpenCL™ Applications - User Manual

Intel OpenCL™ Platform

Intel provides OpenCL™ 1.2 and OpenCL 2.0 platforms to seamlessly access the compute power of such OpenCL devices as Intel CPU and Intel® Processor Graphics. All devices are compliant with the OpenCL 1.2 specification. Intel® Core™ M processors (previously Broadwell) support OpenCL 2.0 across both CPU and GPU OpenCL devices. To get the OpenCL Runtime, refer to the page at https://software.intel.com/en-us/articles/opencl-drivers.

Intel SDK for OpenCL Applications also provides an experimental OpenCL 2.0 platform, which enables you to develop OpenCL 2.0 applications on non Intel Core M systems with an alternative OpenCL 2.0 platform that contains CPU device only. For details, refer to the User Manual section about the OpenCL 2.0 experimental plaform at https://software.intel.com/en-us/node/530840.

Intel's implementation of the OpenCL standard provides support for the Installable Client Driver (ICD), which enables different OpenCL implementations to coexist on the same system. ICD also enables applications to select between OpenCL implementations at run time.

Refer to the following table for information on Intel OpenCL platform ID.

Property	Platform
CL_PLATFORM_NAME	Intel(R) OpenCL
CL_PLATFORM_VENDOR	Intel(R) Corporation
CL_PLATFORM_VERSION	OpenCL 2.0 (on Intel® Core™ M) OpenCL 1.2 (on other supported systems)
OpenCL Devices	CPU and GPU

With installation of the Intel Graphics driver you install the Intel OpenCL platform that enables the following OpenCL devices:

- CL_DEVICE_TYPE_GPU – Intel Graphics
- CL_DEVICE_TYPE_CPU – Intel processor

OpenCL Platform with Intel® Graphics

To enable Intel implementation of the OpenCL software technology on systems with Intel Processor Graphics, install the Intel Graphics driver available at Intel Download Center or with Intel Driver Update Utility.

The following table provides information on the OpenCL software technology version support on Intel Architecture processors:

Intel® Processor Graphics Device	Support on Windows*		Support on Android* OpenCL 1.2 Platform
	OpenCL 1.2 Platform	OpenCL 2.0 Platform	
Intel® HD Graphics 5300 and higher for Intel® Core™ M Processors	✗	✔	✗

4th Generation Intel® Core™ Processors with Intel® Iris™ Pro Graphics 5200	✔	✗	✗
4th Generation Intel® Core™ Processors with Intel® Iris™ Graphics 5100	✔	✗	✗
4th Generation Intel® Core™ Processors with Intel® HD Graphics 5000/4600/4400/4200	✔	✗	✗
Intel® Pentium® Processor 3558U/3560M/3561Y/G3220/G3220T/G3240/ G3240T/G3250/G3250T/G3258/G3320TE/G3420/ G3420T/G3430G3440/G3440T/G3450/G3450T/ G3460 with Intel® HD Graphics	✔	✗	✗
Intel® Pentium® Processor 3550M/3556U/ 3560Y with Intel® HD Graphics	✔	✗	✗
Intel® Celeron® Processor 2957U/2961Y/2970M/2981U/G1820/G1820T/ G1820TE/G1830/G1840/G1840T/G1850 with Intel® HD Graphics	✔	✗	✗
Intel® Celeron® Processor 2000E/2002E/2950M/2955U/2980U with Intel® HD Graphics	✔	✗	✗
Intel(R) Atom(TM) Processor z3700 series	✔	✗	✔

See Also

OpenCL™ Installable Client Driver

OpenCL™ Installable Client Driver

About the Installable Client Driver

OpenCL™ Installable Client Driver (ICD) enables different OpenCL implementations to coexist on the same system. ICD also enables applications to select between OpenCL implementations at run time.

You should select the OpenCL platform for use in your application. If several OpenCL platforms exist in the system, use the `clGetPlatformIDs` and `clGetPlatformInfo` functions to query and select the OpenCL platform. See section "Querying OpenCL Platform ID" for more information.

Installable Client Driver on Windows* OS

By default the environment variable `INTELOCLSDKROOT` is automatically added to the system during installation and points to the OpenCL Runtime installation directory. This directory is also automatically added to the system `PATH` environment. For more information, please see the OpenCL Runtime Release Notes.

To work with the OpenCL runtime, an application should link to the OpenCL Installable Client Driver (ICD) import library, presented as the `OpenCL.lib` file. The library resides in the `$(INTELOCLSDKROOT)\lib\` directory, under `x64` or `x86` subdirectory.

See Also

Querying OpenCL™ Platform ID

Installable Client Driver on Android* OS

To enable the ICD on the target Android* device, make sure the `intel.icd` file exists in the `/system/vendor/Khronos/OpenCL/vendors` folder. The file content is `libintelocl.so`. See the SDK release notes for more information on installing the OpenCL runtime on Android devices.

See Also

Querying OpenCL™ Platform ID

Querying OpenCL™ Platform ID

The following example shows how to query the OpenCL platforms to get the platform ID:

```
cl_platform_id * platforms = NULL;
        char vendor_name[128] = {0};
        cl_uint num_platforms = 0;
// get number of available platforms
        cl_int err = clGetPlatformIDs(0, NULL, & num_platforms);
```

```
        if (CL_SUCCESS != err)
        {
                // handle error
        }
        platforms = (cl_platform_id*)malloc(
                              sizeof(cl_platform)* num_platforms);
        if (NULL == platforms)
        {
                // handle error
        }
        err = clGetPlatformIDs(num_platforms, platforms, NULL);
        if (CL_SUCCESS != err)
        {
                // handle error
        }
        for (cl_uint ui=0; ui< num_platforms; ++ui)
        {
                err = clGetPlatformInfo(platforms[ui],
                              CL_PLATFORM_VENDOR,
                              128 * sizeof(char),
                              vendor_name,
                              NULL);
                if (CL_SUCCESS != err)
                {
                        // handle error
                }
                if (vendor_name != NULL)
                {
                        if (!strcmp(vendor_name, "Intel(R) Corporation"))
                        {
                                return platforms[ui];
                        }
                }
```

```
        }

        // handle error
```

Extensions and Features on GPU

Supported Extensions on GPU

The following OpenCL™ extensions are supported on the Intel® Processor Graphics OpenCL device. For details about the extensions, see Khronos* Extensions Specification.

Extension	Supported on Windows*		Supported on Android* with OpenCL 1.2 Platform	Description
	OpenCL 1.2 Platform	OpenCL 2.0 Platform		
cl_intel_accelerator	✔	✔	✗	Abstraction for domain-specific acceleration engines in the OpenCL™ runtime.
cl_intel_ctz	✔	✔	✗	Adds new built-in kernel function. ctz functions returns the count of trailing 0-bits.
cl_intel_d3d11_nv12_media_sharing	✔	✔	✗	Extends cl_khr_d3d11_sharing interoperability with planar surface formats for media applications.
cl_intel_dx9_media_sharing	✔	✔	✗	Provides interoperability between OpenCL software technology and selected adapter APIs.
cl_intel_motion_estimation	✔	✔	✗	A set of host-callable functions for frame-based motion estimation. This extension depends on the OpenCL 1.2 built-in kernel infrastructure and on the accelerator extension.
cl_khr_3d_image_writes	✔	✔	✔	Enables writes to 3D image memory objects.
cl_khr_byte_addressable_store	✔	✔	✔	Removes restrictions on byte-addressable stores.
cl_khr_d3d10_sharing	✔	✔	✗	Enables sharing of OpenCL and Microsoft DirectX* 10 API resources.

cl_khr_d3d11_sharing	✔	✔	✗	Enables sharing of OpenCL and DirectX 11 API resources.
cl_khr_depth_images	✔	✔	✔	Adds support for depth images in OpenCL image.
cl_khr_dx9_media_sharing	✔	✔	✗	Provides interoperability between OpenCL software technology and selected adapter APIs.
cl_khr_gl_depth_images	✔	✔	✗	Supports OpenCL image to be created from an OpenGL depth or depth-stencil texture.
cl_khr_gl_event	✔	✔	✗	Enables sharing memory objects with OpenGL or OpenGL ES buffers, texture and render bugger objects on Microsoft Windows OS only.
cl_khr_gl_msaa_sharing	✔	✔	✗	Enables MSAA support.
cl_khr_gl_sharing	✔	✔	✗	Enables OpenCL context creation from an OpenGL context or share group on Microsoft Windows operating systems only.
cl_khr_global_int32_base_atomics	✔	✔	✔	Implements atomic operations on 32-bit signed and unsigned integers to locations in __global memory.
cl_khr_global_int32_extended_atomics	✔	✔	✔	Implements atomic operations on 32-bit signed and unsigned integers to locations in __global memory.
cl_khr_icd	✔	✔	✔	Enables OpenCL Installable Client Driver.
cl_khr_image2d_from_buffer	✗	✔	✗	Creating 2D images from OpenCL Buffer.
cl_khr_local_int32_base_atomics	✔	✔	✔	Implement atomic operations on 32-bit signed and unsigned integers to locations in

				__local memory.
cl_khr_local_int32_extended_atomics	✔	✔	✔	Implements atomic operations on 32-bit signed and unsigned integers to locations in __local memory.
cl_khr_mipmap_image	✗	✔	✗	Enables creation of the mip-mapped OpenCL images and sharing the mip-mapped textures from OpenGL.
cl_khr_mipmap_image_writes	✗	✔	✗	Extends cl_khr_mipmap_image by adding write_image* built-in kernel functions to the mip-mapped images.
cl_khr_spir	✔	✔	✗	Enables creating OpenCL program objects from a Standard Portable Intermediate Representation (SPIR) instance.

Supported Features on GPU

Intel implementation of the OpenCL™ standard provides support for the following optional features on Intel® Processor Graphics OpenCL device.

Feature	**Supported on Windows***		**Supported on Android* with OpenCL 1.2 Platform**	**Description**
	OpenCL 1.2 Platform	OpenCL 2.0 Platform		
Shared Context	✔	✔	✔	Enables OpenCL memory and events to share different device facilities.
Image Support	✔	✔	✔	Enables 1D images and 1D/2D image arrays support. For full list of supported image formats – see chapter "Appendix A - Supported Image Formats" of this guide. If the device supports images, CL_DEVICE_IMAGE_SUPPORT is CL_TRUE.
Writing to the 3D Image Memory Object	✔	✔	✔	The Intel Graphics device implements the cl_khr_3d_image_writes extension to support writes to a 3D image memory object.

Microsoft DirectX* 9 Media Sharing	✔	✔	✗	Enables sharing of the OpenCL and DirectX 9 API resources.
Microsoft DirectX 10 Media Sharing	✔	✔	✗	Enables sharing of the OpenCL and DirectX 10 API resources.
Microsoft DirectX 11 Media Sharing	✔	✔	✗	Enables sharing of the OpenCL and DirectX 11 API resources.
OpenGL* Sharing	✔	✔	✗	Enables sharing of the OpenGL and the OpenCL resources.
MSAA Sharing of OpenCL and OpenGL	✔	✔	✗	Enables creating OpenCL images from OpenGL multi-sampled textures.
Interoperability with Media and Graphics APIs	✔	✔	✗	Provides interoperability with other graphics and media APIs such as Microsoft DirectX, OpenGL, and the Intel® Media SDK.
C11 Atomics	✗	✔	✗	Enables assignments in one work-item to be visible to other work-items in a work-group, across work-groups executing on a device or for sharing data between the OpenCL device and host.
Shared Virtual Memory	✗	✔	✗	Enables host and device kernels to directly share complex, pointer-containing data structures such as trees and linked lists, providing significant programming flexibility and eliminating costly data transfers between host and devices.
Dynamic Parallelism (Device Enqueue)	✗	✔	✗	Device kernels can enqueue kernels to the same device with no host interaction, enabling flexible work scheduling paradigms and avoiding the need to transfer execution control and data between the device and host, often significantly offloading host processor bottlenecks.
Generic Address Space	✗	✔	✗	Functions can be written without specifying a named address space for arguments, especially useful for those arguments that are declared to be a pointer to a type, eliminating the need for multiple functions to be written for each named address space used in an application.
sRGB Images	✗	✔	✗	Provides embedded sRGB image format support. The new feature handles conversion from sRGB into RGB values and speeds up both the development time and the kernel performance.

RW Images	✗	✔	✗	Enables kernels to read from and write to the same image.
Pipes	✗	✔	✗	Pipes are memory objects that store data organized as a FIFO and OpenCL 2.0 provides built-in functions for kernels to read from or write to a pipe, providing straightforward programming of pipe data structures that can be highly optimized by OpenCL implementers.
Non-Uniform Work-groups	✗	✔	✗	Enables the OpenCL 2.0 runtime to divide an NDRange in a way that produces non-uniform work-group sizes in any dimension.
New Work-group Built-in Functions	✗	✔	✗	Introduceds work-group functions, which are built-ins that provide popular parallel primitives that operate at the workgroup level: value broadcast, reduce, and scan, plus two built-ins that evaluate boolean operation result over entire workgroup.
Mipmaps (KHR Optional Extension)	✗	✔	✗	Enables creation of OpenCL images from a mip-mapped or a multi-sampled OpenGL texture for improved OpenGL interoperability.

For full list of supported images formats – see chapter "Supported Image Formats" of this guide.

See Also

Intel Threading Building Blocks website
Supported Image Formats

Extensions and Features on CPU

Supported Extensions on CPU

The following OpenCL™ extensions are supported on Intel CPU OpenCL device:

Extension	Supported on Windows*		Supported on Android* with OpenCL 1.2 Platform	Description
	OpenCL 1.2 Platform	OpenCL 2.0 Platform		
cl_intel_dx9_media_sharing	✔	✔	✗	Provides interoperability between OpenCL software technology and selected adapter APIs.
cl_intel_exec_by_local_thread	✔	✔	✔	Enables OpenCL commands execution in a single-threaded manner, using the calling thread to perform the actual execution.
cl_khr_3d_image_writes	✔	✔	✔	Enables writes to 3D image memory objects.
cl_khr_byte_addressable_store	✔	✔	✔	Removes restrictions on byte-addressable stores.
cl_khr_d3d11_sharing	✔	✔	✗	Enables sharing of OpenCL and DirectX 11 API resources.
cl_khr_depth_images	✔	✔	✔	Adds support for depth images in OpenCL image.
cl_khr_dx9_media_sharing	✔	✔	✗	Provides interoperability between OpenCL software technology and selected adapter APIs.
cl_khr_fp64	✔	✔	✗	Enables double-precision floating point support.
cl_khr_gl_sharing	✔	✔	✗	Enables OpenCL context creation from an

				OpenGL context or share group on Microsoft Windows operating systems only.
cl_khr_global_int32_base_atomics	✔	✔	✔	Implements atomic operations on 32-bit signed and unsigned integers to locations in __global memory.
cl_khr_global_int32_extended_atomics	✔	✔	✔	Implements atomic operations on 32-bit signed and unsigned integers to locations in __global memory.
cl_khr_icd	✔	✔	✔	Enables OpenCL Installable Client Driver.
cl_khr_image2d_from_buffer	✗	✔	✗	Creating 2D images from OpenCL Buffer.
cl_khr_local_int32_base_atomics	✔	✔	✔	Implement atomic operations on 32-bit signed and unsigned integers to locations in __local memory.
cl_khr_local_int32_extended_atomics	✔	✔	✔	Implements atomic operations on 32-bit signed and unsigned integers to locations in __local memory.
cl_khr_spir	✔	✔	✗	Enables creating OpenCL program objects from a Standard Portable Intermediate Representation (SPIR) instance.

OpenCL 2.0 experimental development environment provides support to all of the OpenCL 2.0 features available with the Intel Core M processors (formerly Broadwell).

Supported Features on CPU

Intel implementation of the OpenCL™ standard provides support for the following optional features on the Intel CPU OpenCL device.

Feature	Supported on Windows*	Supported on	Description

	OpenCL 1.2 Platform	OpenCL 2.0 Platform	**Android* with OpenCL 1.2 Platform**	
Shared Context	✔	✔	✔	Enables OpenCL memory and events to share different device facilities.
Image Support	✔	✔	✔	Enables 1D images and 1D/2D image arrays support. For full list of supported image formats – see chapter "Supported Image Formats" of this manual. If the device supports images, `CL_DEVICE_IMAGE_SUPPORT` is `CL_TRUE`.
Microsoft DirectX* 9 Media Sharing	✔	✔	✗	Enables sharing of the OpenCL and DirectX 9 API resources.
Microsoft DirectX 11 Media Sharing	✔	✔	✗	Enables sharing of the OpenCL and DirectX 11 API resources.
OpenGL* Sharing	✔	✔	✗	Enables sharing of the OpenGL and the OpenCL resources.
Double-Precision Floating Point	✔	✔	✗	Provides support for double-precision floating-point.
Intel Immediate Command Execution	✔	✔	✗	Enables you to execute OpenCL commands in a single-threaded manner.
Final Kernel Binary Save/Load	✔	✔	✗	Facilitates CPU compile time saving. Intel implementation of the existing OpenCL API for CPU is enhanced to retrieve and inject binaries of the programs that are compiled and optimized for CPU. Namely, the binary string returned by invoking `clGetProgramInfo()` with parameter `CL_PROGRAM_BINARIES` contains a target-specific compiled binary, after building the program for CPU device. This binary is persistent, and you can save it to disk. If you invoke `clCreateProgramWithBinary()` with such a binary, it will be used without compiling the program. Note: Ensure that a correct binary is provided to `clCreateProgramWithBinary()`.
Out-of-order Execution	✔	✔	✗	Enables support of an out-of-order execution model for kernels and memory objects in the device command queue

				(CL_QUEUE_OUT_OF_ORDER_EXEC_MODE_ENABLE property of a command-queue).
Native Kernels	✔	✔	✗	Enables native kernel execution. The CPU device supports execution of native kernels (CL_EXEC_NATIVE_KERNEL option of the CL_DEVICE_EXECUTION_CAPABILITIES property of device information). Access native kernels through a host function pointer. Queued native kernels for execution along with OpenCL kernels on a device and share memory objects with OpenCL kernels. For example, these native kernels could be functions defined in application code or exported from a library.
Implicit CPU Vectorization	✔	✔	✔	Aims to merge together the execution of several work items, utilizing the Intel vector instruction set and extends the utilization of the vector unit when moving from one generation to another.
Scalable Threading System	✔	✔	✗	The system is based on the Intel® Threading Building Blocks (Intel® TBB). This runtime feature enables the OpenCL applications to seamlessly utilize the multicore CPU.
C11 Atomics	✗	✔	✗	Enables assignments in one work-item to be visible to other work-items in a work-group, across work-groups executing on a device or for sharing data between the OpenCL device and host.
Shared Virtual Memory	✗	✔	✗	Enables host and device kernels to directly share complex, pointer-containing data structures such as trees and linked lists, providing significant programming flexibility and eliminating costly data transfers between host and devices.
Dynamic Parallelism (Device Enqueue)	✗	✔	✗	Device kernels can enqueue kernels to the same device with no host interaction, enabling flexible work scheduling paradigms and avoiding the need to transfer execution control and data between the device and host, often significantly offloading host processor bottlenecks.
Generic Address Space	✗	✔	✗	Functions can be written without specifying a named address space for arguments, especially useful for those arguments that are declared to be a pointer to a type, eliminating the need for multiple functions to be written for each named address space used in an application.
sRGB Images	✗	✔	✗	Provides embedded sRGB image format support. The new feature handles conversion from sRGB into RGB values and speeds up both the development time and the kernel performance.
Pipes	✗	✔	✗	Pipes are memory objects that store data organized

				as a FIFO and OpenCL 2.0 provides built-in functions for kernels to read from or write to a pipe, providing straightforward programming of pipe data structures that can be highly optimized by OpenCL implementers.
Non-Uniform Work-groups	✗	✔	✗	Enables the OpenCL 2.0 runtime to divide an NDRange in a way that produces non-uniform work-group sizes in any dimension.
New Work-group Built-in Functions	✗	✔	✗	Introduceds work-group functions, which are built-ins that provide popular parallel primitives that operate at the workgroup level: value broadcast, reduce, and scan, plus two built-ins that evaluate boolean operation result over entire workgroup.
Mipmaps (KHR Optional Extension)	✗	✔	✗	Enables creation of OpenCL images from a mip-mapped or a multi-sampled OpenGL texture for improved OpenGL interoperability.

For full list of supported images formats – see chapter "Supported Image Formats" of this guide.

See Also

Intel Threading Building Blocks website
Supported Image Formats

Notes on Features

Shared Context

Creating Shared Context

Intel implementation of the OpenCL™ standard supports context for multiple devices, also called “shared context”. An OpenCL context of an Intel® processor and Intel Graphics device enables memory and events to share different device facilities. This feature eases development of workloads that run across the platform (CPU and GPU).

To create a shared context for all devices:

```
shared_context = clCreateContextFromType(prop, CL_DEVICE_TYPE_ALL, …);
```

Do not specify `CL_DEVICE_TYPE_ALL` if the application targets a single device context, either Intel CPU or Intel Graphics.

To create a shared context for a single device, specify `CL_DEVICE_TYPE_CPU` or `CL_DEVICE_TYPE_GPU` explicitly:

```
cl_device_id devices[2] = {cpuDeviceId , gpuDeviceId};

cl_context shared_context = clCreateContext(prop, 1, devices, …);
```

For more information on the functionality of a shared context, self-management and application-level management, see the OpenCL™ 1.2 specification.

You do not need to worry about memory object mirroring or migration between different context devices, just avoid concurrent “Write” access to the same memory object by the different devices, as stated in the OpenCL 1.2 Specification.

The following extensions are not supported in the shared context:

- `cl_khr_gl_sharing`
- `cl_khr_d3d10_sharing`
- `cl_khr_d3d11_sharing`
- `cl_intel_dx9_media_sharing`
- `cl_intel_d3d11_nv12_media_sharing`

See Also

OpenCL 1.2 Specification at http://www.khronos.org/registry/cl/specs/opencl-1.2.pdf

Resource Sharing

OpenCL™ memory objects, created on a shared context, are shared among the context devices. Applications do not need to copy them (using a "Read" or "Write" operation) between the context devices to process on the target device.

Intel OpenCL devices use true resource sharing in a shared context. There is no hidden memory copy when processing the memory object on *different* devices. However, there might still be a copy on `clEnqueueMapBuffer[Image]` or `clEnqueueUnmapMemObject` of the memory objects created with the `CL_MEM_USE_HOST_PTR` flag set.

The *OpenCL™ Optimization Guide for Visual Computing Systems* provides more information on how to avoid memory synchronization overhead with the host application.

See Also

Supported OpenCL Extensions
Supported Features Summary
OpenCL™ Optimization Guide for Visual Computing Systems

Interoperability with Media and Graphics APIs

OpenCL™ Runtime installed with the Intel® Graphics driver provides interoperability with other graphics and media APIs such as Microsoft DirectX*, OpenGL*, and the Intel Media SDK. Graphics and Media interoperability enables the applications that use these APIs to benefit from true surface sharing and zero copy on the Intel Graphics OpenCL device, when using according to condition in the table below:

Extension	Condition
DirectX 9 Media SurfaceSharing	Provide a non-`NULL` `pSharedHandle` to `clCreateFromDX9MediaSurfaceKHR` to implement true sharing.
DirectX 10 and 11 Sharing Memory Objects	Create a resource with `D3D10_RESOURCE_MISC_SHARED` and `D3D11_RESOURCE_MISC_SHARED` flag specified to implement true sharing.
OpenGL Sharing Memory Objects	Depends on your hardware state. No specific way for the application to implement true sharing.

For information on `pSharedHandle`, see the “Feature Summary (Direct3D* 9 for Windows Vista*)” article.

Use the DirectX 9 API also for interoperability with Intel Media SDK.

For more details on how to take advantage of the interoperability between OpenCL and Intel Media SDK, see the Intel Media SDK Interoperability code sample.

For more information on `D3D10_RESOURCE_MISC_SHARED` and `D3D11_RESOURCE_MISC_SHARED`, see “Feature Summary (Direct3D 9 for Windows Vista)” article.

See Also

Intel Media SDK website
Interoperability with Intel Media SDK sample
Feature Summary (Direct3D 9 for Windows Vista) at http://msdn.microsoft.com/en-us/library/windows/desktop/bb219800(v=vs.85).aspx

Supported Image Formats

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804

Read-Only Surface Formats

The following is the list of read-only surface formats supported with both OpenCL 1.2 platform and OpenCL 2.0 platform (available with Intel® Core™ M processors).

cl_channel_order	cl_channel_type	GPU	CPU
CL_RGBA	CL_UNORM_INT8	Yes	Yes
CL_RGBA	CL_UNORM_INT16	Yes	Yes
CL_RGBA	CL_SIGNED_INT8	Yes	Yes
CL_RGBA	CL_SIGNED_INT16	Yes	Yes
CL_RGBA	CL_SIGNED_INT32	Yes	Yes
CL_RGBA	CL_UNSIGNED_INT8	Yes	Yes
CL_RGBA	CL_UNSIGNED_INT16	Yes	Yes
CL_RGBA	CL_UNSIGNED_INT32	Yes	Yes
CL_RGBA	CL_HALF_FLOAT	Yes	Yes
CL_RGBA	CL_FLOAT	Yes	Yes
CL_RGBA	CL_SNORM_INT8	Yes	Yes
CL_RGBA	CL_SNORM_INT16	Yes	Yes
CL_BGRA	CL_UNORM_INT8	Yes	Yes
CL_R	CL_FLOAT	Yes	Yes

CL_R	CL_UNORM_INT8	**Yes**	**Yes**
CL_R	CL_UNORM_INT16	**Yes**	**Yes**
CL_R	CL_SIGNED_INT8	**Yes**	**Yes**
CL_R	CL_SIGNED_INT16	**Yes**	**Yes**
CL_R	CL_SIGNED_INT32	**Yes**	**Yes**
CL_R	CL_UNSIGNED_INT8	**Yes**	**Yes**
CL_R	CL_UNSIGNED_INT16	**Yes**	**Yes**
CL_R	CL_UNSIGNED_INT32	**Yes**	**Yes**
CL_R	CL_HALF_FLOAT	**Yes**	**Yes**
CL_R	CL_SNORM_INT8	**Yes**	**Yes**
CL_R	CL_SNORM_INT16	**Yes**	**Yes**
CL_INTENSITY	CL_UNORM_INT8	**Yes**	**Yes**
CL_INTENSITY	CL_UNORM_INT16	**Yes**	**Yes**
CL_INTENSITY	CL_HALF_FLOAT	**Yes**	**Yes**
CL_INTENSITY	CL_FLOAT	**Yes**	**Yes**
CL_LUMINANCE	CL_UNORM_INT8	**Yes**	**Yes**
CL_LUMINANCE	CL_UNORM_INT16	**Yes**	**Yes**
CL_LUMINANCE	CL_HALF_FLOAT	**Yes**	**Yes**
CL_LUMINANCE	CL_FLOAT	**Yes**	**Yes**
CL_A	CL_UNORM_INT8	**Yes**	**Yes**
CL_A	CL_UNORM_INT16	**Yes**	**Yes**
CL_A	CL_HALF_FLOAT	**Yes**	**Yes**
CL_A	CL_FLOAT	**Yes**	**Yes**
CL_RG	CL_UNORM_INT8	**Yes**	**Yes**
CL_RG	CL_UNORM_INT16	**Yes**	**Yes**

CL_RG	CL_SIGNED_INT8	**Yes**	**Yes**
CL_RG	CL_SIGNED_INT16	**Yes**	**Yes**
CL_RG	CL_SIGNED_INT32	**Yes**	**Yes**
CL_RG	CL_UNSIGNED_INT8	**Yes**	**Yes**
CL_RG	CL_UNSIGNED_INT16	**Yes**	**Yes**
CL_RG	CL_UNSIGNED_INT32	**Yes**	**Yes**
CL_RG	CL_HALF_FLOAT	**Yes**	**Yes**
CL_RG	CL_FLOAT	**Yes**	**Yes**
CL_RG	CL_SNORM_INT8	**Yes**	**Yes**
CL_RG	CL_SNORM_INT16	**Yes**	**Yes**
CL_sRGBA	CL_UNORM_INT8	**Yes**	**Yes**
CL_sBGRA	CL_UNORM_INT8	**Yes**	**Yes**
CL_DEPTH	CL_FLOAT	**Yes**	**Yes**
CL_DEPTH	CL_UNORM_INT16	**Yes**	**Yes**
CL_DEPTH_STENCIL	CL_UNORM_INT24	**Yes**	No
CL_DEPTH_STENCIL	CL_FLOAT	**Yes**	No

Write-Only Surface Formats

The following is the list of write-only surface formats supported with both OpenCL 1.2 platform and OpenCL 2.0 platform (available with Intel® Core™ M processors).

cl_channel_order	cl_channel_type	GPU	CPU
CL_RGBA	CL_UNORM_INT8	**Yes**	**Yes**
CL_RGBA	CL_UNORM_INT16	**Yes**	**Yes**
CL_RGBA	CL_SIGNED_INT8	**Yes**	**Yes**
CL_RGBA	CL_SIGNED_INT16	**Yes**	**Yes**
CL_RGBA	CL_SIGNED_INT32	**Yes**	**Yes**

CL_RGBA	CL_UNSIGNED_INT8	**Yes**	**Yes**
CL_RGBA	CL_UNSIGNED_INT16	**Yes**	**Yes**
CL_RGBA	CL_UNSIGNED_INT32	**Yes**	**Yes**
CL_RGBA	CL_HALF_FLOAT	**Yes**	**Yes**
CL_RGBA	CL_FLOAT	**Yes**	**Yes**
CL_RGBA	CL_SNORM_INT8	**Yes**	**Yes**
CL_RGBA	CL_SNORM_INT16	**Yes**	**Yes**
CL_BGRA	CL_UNORM_INT8	**Yes**	**Yes**
CL_R	CL_FLOAT	**Yes**	**Yes**
CL_R	CL_UNORM_INT8	**Yes**	**Yes**
CL_R	CL_UNORM_INT16	**Yes**	**Yes**
CL_R	CL_SIGNED_INT8	**Yes**	**Yes**
CL_R	CL_SIGNED_INT16	**Yes**	**Yes**
CL_R	CL_SIGNED_INT32	**Yes**	**Yes**
CL_R	CL_UNSIGNED_INT8	**Yes**	**Yes**
CL_R	CL_UNSIGNED_INT16	**Yes**	**Yes**
CL_R	CL_UNSIGNED_INT32	**Yes**	**Yes**
CL_R	CL_HALF_FLOAT	**Yes**	**Yes**
CL_R	CL_SNORM_INT8	**Yes**	**Yes**
CL_R	CL_SNORM_INT16	**Yes**	**Yes**
CL_INTENSITY	CL_UNORM_INT8	No	**Yes**
CL_INTENSITY	CL_UNORM_INT16	No	**Yes**
CL_INTENSITY	CL_HALF_FLOAT	No	**Yes**
CL_INTENSITY	CL_FLOAT	No	**Yes**
CL_LUMINANCE	CL_UNORM_INT8	No	**Yes**

CL_LUMINANCE	CL_UNORM_INT16	No	**Yes**
CL_LUMINANCE	CL_HALF_FLOAT	No	**Yes**
CL_LUMINANCE	CL_FLOAT	No	**Yes**
CL_A	CL_UNORM_INT8	**Yes**	**Yes**
CL_A	CL_UNORM_INT16	No	**Yes**
CL_A	CL_HALF_FLOAT	No	**Yes**
CL_A	CL_FLOAT	No	**Yes**
CL_RG	CL_UNORM_INT8	**Yes**	**Yes**
CL_RG	CL_UNORM_INT16	**Yes**	**Yes**
CL_RG	CL_SIGNED_INT8	**Yes**	**Yes**
CL_RG	CL_SIGNED_INT16	**Yes**	**Yes**
CL_RG	CL_SIGNED_INT32	**Yes**	**Yes**
CL_RG	CL_UNSIGNED_INT8	**Yes**	**Yes**
CL_RG	CL_UNSIGNED_INT16	**Yes**	**Yes**
CL_RG	CL_UNSIGNED_INT32	**Yes**	**Yes**
CL_RG	CL_HALF_FLOAT	**Yes**	**Yes**
CL_RG	CL_FLOAT	**Yes**	**Yes**
CL_RG	CL_SNORM_INT8	**Yes**	**Yes**
CL_RG	CL_SNORM_INT16	**Yes**	**Yes**
CL_DEPTH	CL_FLOAT	**Yes**	**Yes**
CL_DEPTH	CL_UNORM_INT16	**Yes**	**Yes**

Read or Write Surface Formats

The following is the list of read or write surface formats supported with both OpenCL 1.2 platform and OpenCL 2.0 platform (available with Intel® Core™ M processors).

cl_channel_order	cl_channel_type	GPU	CPU

CL_RGBA	CL_UNORM_INT8	**Yes**	**Yes**
CL_RGBA	CL_UNORM_INT16	**Yes**	**Yes**
CL_RGBA	CL_SIGNED_INT8	**Yes**	**Yes**
CL_RGBA	CL_SIGNED_INT16	**Yes**	**Yes**
CL_RGBA	CL_SIGNED_INT32	**Yes**	**Yes**
CL_RGBA	CL_UNSIGNED_INT8	**Yes**	**Yes**
CL_RGBA	CL_UNSIGNED_INT16	**Yes**	**Yes**
CL_RGBA	CL_UNSIGNED_INT32	**Yes**	**Yes**
CL_RGBA	CL_HALF_FLOAT	**Yes**	**Yes**
CL_RGBA	CL_FLOAT	**Yes**	**Yes**
CL_RGBA	CL_SNORM_INT8	**Yes**	**Yes**
CL_RGBA	CL_SNORM_INT16	**Yes**	**Yes**
CL_BGRA	CL_UNORM_INT8	**Yes**	**Yes**
CL_R	CL_FLOAT	**Yes**	**Yes**
CL_R	CL_UNORM_INT8	**Yes**	**Yes**
CL_R	CL_UNORM_INT16	**Yes**	**Yes**
CL_R	CL_SIGNED_INT8	**Yes**	**Yes**
CL_R	CL_SIGNED_INT16	**Yes**	**Yes**
CL_R	CL_SIGNED_INT32	**Yes**	**Yes**
CL_R	CL_UNSIGNED_INT8	**Yes**	**Yes**
CL_R	CL_UNSIGNED_INT16	**Yes**	**Yes**
CL_R	CL_UNSIGNED_INT32	**Yes**	**Yes**
CL_R	CL_HALF_FLOAT	**Yes**	**Yes**
CL_R	CL_SNORM_INT8	**Yes**	**Yes**
CL_R	CL_SNORM_INT16	**Yes**	**Yes**

CL_INTENSITY	CL_UNORM_INT8	No	**Yes**
CL_INTENSITY	CL_UNORM_INT16	No	**Yes**
CL_INTENSITY	CL_HALF_FLOAT	No	**Yes**
CL_INTENSITY	CL_FLOAT	No	**Yes**
CL_LUMINANCE	CL_UNORM_INT8	No	**Yes**
CL_LUMINANCE	CL_UNORM_INT16	No	**Yes**
CL_LUMINANCE	CL_HALF_FLOAT	No	**Yes**
CL_LUMINANCE	CL_FLOAT	No	**Yes**
CL_A	CL_UNORM_INT8	**Yes**	**Yes**
CL_A	CL_UNORM_INT16	No	**Yes**
CL_A	CL_HALF_FLOAT	No	**Yes**
CL_A	CL_FLOAT	No	**Yes**
CL_RG	CL_UNORM_INT8	**Yes**	**Yes**
CL_RG	CL_UNORM_INT16	**Yes**	**Yes**
CL_RG	CL_SIGNED_INT8	**Yes**	**Yes**
CL_RG	CL_SIGNED_INT16	**Yes**	**Yes**
CL_RG	CL_SIGNED_INT32	**Yes**	**Yes**
CL_RG	CL_UNSIGNED_INT8	**Yes**	**Yes**
CL_RG	CL_UNSIGNED_INT16	**Yes**	**Yes**
CL_RG	CL_UNSIGNED_INT32	**Yes**	**Yes**
CL_RG	CL_HALF_FLOAT	**Yes**	**Yes**
CL_RG	CL_FLOAT	**Yes**	**Yes**
CL_RG	CL_SNORM_INT8	**Yes**	**Yes**
CL_RG	CL_SNORM_INT16	**Yes**	**Yes**
CL_DEPTH	CL_FLOAT	**Yes**	**Yes**

CL_DEPTH	CL_UNORM_INT16	**Yes**	**Yes**

OpenCL 2.0 Write-Only Surface Formats

The following is the list of read and write surface formats supported with OpenCL 2.0 platform available with Intel® Core™ M processors.

cl_channel_order	cl_channel_type	GPU	CPU
CL_RGBA	CL_UNORM_INT8	**Yes**	**Yes**
CL_RGBA	CL_UNORM_INT16	**Yes**	**Yes**
CL_RGBA	CL_SIGNED_INT8	**Yes**	**Yes**
CL_RGBA	CL_SIGNED_INT16	**Yes**	**Yes**
CL_RGBA	CL_SIGNED_INT32	**Yes**	**Yes**
CL_RGBA	CL_UNSIGNED_INT8	**Yes**	**Yes**
CL_RGBA	CL_UNSIGNED_INT16	**Yes**	**Yes**
CL_RGBA	CL_UNSIGNED_INT32	**Yes**	**Yes**
CL_RGBA	CL_HALF_FLOAT	**Yes**	**Yes**
CL_RGBA	CL_FLOAT	**Yes**	**Yes**
CL_RGBA	CL_SNORM_INT8	**Yes**	**Yes**
CL_RGBA	CL_SNORM_INT16	**Yes**	**Yes**
CL_BGRA	CL_UNORM_INT8	**Yes**	**Yes**
CL_R	CL_FLOAT	**Yes**	**Yes**
CL_R	CL_UNORM_INT8	**Yes**	**Yes**
CL_R	CL_UNORM_INT16	**Yes**	**Yes**
CL_R	CL_SIGNED_INT8	**Yes**	**Yes**
CL_R	CL_SIGNED_INT16	**Yes**	**Yes**
CL_R	CL_SIGNED_INT32	**Yes**	**Yes**
CL_R	CL_UNSIGNED_INT8	**Yes**	**Yes**

CL_R	CL_UNSIGNED_INT16	**Yes**	**Yes**
CL_R	CL_UNSIGNED_INT32	**Yes**	**Yes**
CL_R	CL_HALF_FLOAT	**Yes**	**Yes**
CL_R	CL_SNORM_INT8	**Yes**	**Yes**
CL_R	CL_SNORM_INT16	**Yes**	**Yes**
CL_INTENSITY	CL_UNORM_INT8	No	**Yes**
CL_INTENSITY	CL_UNORM_INT16	No	**Yes**
CL_INTENSITY	CL_HALF_FLOAT	No	**Yes**
CL_INTENSITY	CL_FLOAT	No	**Yes**
CL_LUMINANCE	CL_UNORM_INT8	No	**Yes**
CL_LUMINANCE	CL_UNORM_INT16	No	**Yes**
CL_LUMINANCE	CL_HALF_FLOAT	No	**Yes**
CL_LUMINANCE	CL_FLOAT	No	**Yes**
CL_A	CL_UNORM_INT8	**Yes**	**Yes**
CL_A	CL_UNORM_INT16	No	**Yes**
CL_A	CL_HALF_FLOAT	No	**Yes**
CL_A	CL_FLOAT	No	**Yes**
CL_RG	CL_UNORM_INT8	**Yes**	**Yes**
CL_RG	CL_UNORM_INT16	**Yes**	**Yes**
CL_RG	CL_SIGNED_INT8	**Yes**	**Yes**
CL_RG	CL_SIGNED_INT16	**Yes**	**Yes**
CL_RG	CL_SIGNED_INT32	**Yes**	**Yes**
CL_RG	CL_UNSIGNED_INT8	**Yes**	**Yes**
CL_RG	CL_UNSIGNED_INT16	**Yes**	**Yes**
CL_RG	CL_UNSIGNED_INT32	**Yes**	**Yes**

CL_RG	CL_HALF_FLOAT	**Yes**	**Yes**
CL_RG	CL_FLOAT	**Yes**	**Yes**
CL_RG	CL_SNORM_INT8	**Yes**	**Yes**
CL_RG	CL_SNORM_INT16	**Yes**	**Yes**
CL_DEPTH	CL_FLOAT	**Yes**	**Yes**
CL_DEPTH	CL_UNORM_INT16	**Yes**	**Yes**

OpenCL™ Build and Linking Options

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804

Preprocessor Options

Option	Description	GPU	CPU
`-D <name>`	Predefines name as a macro, with definition 1.	**Yes**	**Yes**
`-D <name=definition>`	The contents of definition become tokens and are processed as if they appeared during translation phase three in a `#define` directive. The definition is truncated by embedded newline characters.	**Yes**	**Yes**
`-I <dir>`	Adds directory `<dir>` to the list of directories for header files search.	**Yes**	**Yes**

Math Intrinsic Options

Option	Description	GPU	CPU
`-cl-single-precision-constant`	Treats double-precision floating point constant as single-precision constant.	**Yes**	No
`-cl-denorms-are-zero`	This option controls how single-precision and double-precision denormalized numbers are handled. If specified as a build option, single-precision denormalized numbers may be flushed to zero; double-precision denormalized numbers may also be flushed to zero if the optional extension for double-precision is supported.	No	**Yes**
`-cl-fp32-correctly-rounded-divide-sqrt`	This option enables an application to specify that single precision		

	floating-point divide (x/y and 1/x) and `sqrt` used in the program source are correctly rounded	

Optimization Options

Option	Description	GPU	CPU
`-cl-opt-disable`	This option disables all optimizations. Optimizations are enabled by default.	No	**Yes**
`-cl-mad-enable`	Enables `a * b + c` to be replaced by `mad`. Note that `mad` computes `a * b + c` with reduced accuracy.	**Yes**	No
`-cl-no-signed-zeros`	Enables optimizations for floating-point arithmetic that ignore the signedness of zero. IEEE 754 arithmetic specifies the distinct behavior of +0.0 and -0.0 values, which then prohibits simplification of expressions such as x+0.0 or 0.0*x (even with -clfinite- math only). This option implies that the sign of a zero result isn't significant.	**Yes**	No
`-cl-unsafe-math-optimizations`	Enables optimizations for floating-point arithmetic that, • assume that arguments and results are valid • may violate IEEE 754 standard • may violate the OpenCL™ numerical compliance requirements	**Yes**	No
`-cl-finite-math-only`	Enables optimizations for floating-point arithmetic that assume that arguments and results are not `NaNs` or >±∞.	**Yes**	No
`-cl-fast-relaxed-math`	Sets the optimization options `-cl-finite-math-only` and `-cl-unsafe-math-optimizations`, which enables optimizations for floating-point arithmetic that may violate the IEEE 754 standard and the OpenCL™ numerical compliance requirements.	**Yes**	**Yes**
`-cl-uniform-work-group-size`	This requires the global work-size to be multiple of the work-group size specified to `clEnqueueNDRangeKernel`. Enables optimizations that are made possible by this restriction.	**Yes**	**Yes**

Options for Warnings

Option	Description	GPU	CPU
`-w`	Toggles all warning messages.	**Yes**	**Yes**
`-Werror`	Makes all warnings errors.	**Yes**	**Yes**

Options Controlling the OpenCL™ C Version

Option	Description	GPU	CPU
`-cl-std=`	Determine the OpenCL™ C language version to use.	**Yes**	**Yes**